NASA Contractor Report 181961
ICASE Report No. 89-54

# ICASE

## KRYLOV METHODS PRECONDITIONED WITH INCOMPLETELY FACTORED MATRICES ON THE CM-2

Harry Berryman
Joel Saltz
William Gropp
Ravi Mirchandaney



Contract No. NAS1-18605
December 1989

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665-5225

Operated by the Universities Space Research Association



National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

# Krylov Methods Preconditioned with Incompletely Factored Matrices on the CM-2 *

*Harry Berryman*
*Joel Saltz*
*William Gropp*
*Ravi Mirchandaney*
ICASE
NASA Langley Research Center
Hampton, VA 23665
Department of Computer Science
Yale University
New Haven, CT 06520

December 17, 1989



The authors

**Abstract**

In the work presented here, we measured the performance of the components of the key iterative kernel of a preconditioned Krylov space iterative linear system solver. In some sense, these numbers can be regarded as best case timings for these kernels. We timed sweeps over meshes, sparse triangular solves, and inner products on a large three dimensional model problem over a cube shaped domain discretised with a seven point template.

The performance of the CM-2 is highly dependent on the use of very specialised programs. These programs mapped a regular problem domain onto the processor topology in a careful manner and used the optimised local NEWS communications network. We also document rather dramatic deterioration in performance when these ideal conditions no longer apply. A synthetic workload generator was developed to produce and solve a parameterised family of increasingly irregular problems.

Keywords: PARIS programming language, assembly language, (KR)

*This work was supported by the U.S. Office of Naval Research under Grant N00014-86-K-0310 through Yale, by DARPA contract DAA-1101-88-C-0409 through Scientific Computing Associates, and by NASA contract NAS1-18605 while authors Berryman, Salts and Mirchandaney were in residence at ICASE, NASA Langley Research Center

# 1 Overview

In the work described in this paper, we have carefully examined a set of model problems to demonstrate the range of performance that one can anticipate from the CM-2. We first present results that arose from experiments in which we gave the machine and its software every conceivable advantage. We used PARIS, the CM-2 assembly language, to program a computational kernel as it might appear in the iterative portion of a Krylov space iterative linear equation solver preconditioned with an incompletely factored matrix [4], [2], [5], [6]. This kernel was coded with the assumption that the linear system being solve arose from a partial differential equation discretized with a uniform template on a three dimensional mesh. We did not address the issues that arise in considering the tradeoffs between computational rates and rates of convergence that go into deciding what kind of preconditioning to use in a massively parallel Krylov space solver. Attempts to address this issue have been made by [1] and [3].

The performance of the CM-2 was highly dependent on the use of very specialized programs. These programs mapped a regular problem domain onto the processor topology in a careful manner and used the optimized local NEWS communications network. Efficient methods for solving partial differential equations frequently make use of non-uniform grids designed to put the most computational effort where the problem is hardest. An effect of this approach is that the algebraic linear (and non-linear) systems that must eventually be solved are sparse and quite irregular in structure. Careful mapping of workload can be extremely important in obtaining adequate performance from many multiprocessor architectures; mapping is typically straightforward in regular problems with a known structure and is much more complicated for problems with unknown or irregular structures.

In this paper, we document rather dramatic deterioration in performance when ideal conditions no longer apply. A synthetic workload generator was developed to produce and solve a parameterized family of increasingly irregular problems. These problems involved sweeps over meshes. The irregularities were obtained by altering square meshes having nearest neighbor links by replacing varying fractions of these local links with dependencies involving mesh points that were more remote. We also explored the performance implications of using the high level language *lisp to implement a mesh sweep in a way that does not rely on a detailed a-priori knowledge of a problems communication characteristics.

In Section 2, we present best case timings for the matrix vector multiplies, sparse triangular solves, inner products and SAXPYs that constitute the iterative portion of Krylov based programs. In Section 3, we present data which indicates that performance of the CM-2 for regular problems is an extremely sensitive function of 1) problem regularity, 2) problem mapping and 3) *a priori* knowledge of dependency patterns.

# 2 Performance on a Regular Three Dimensional Mesh

In this section we describe the results of experiments that give a best case estimate of the rate with which the CM-2 can carry out the procedures that make up the iterative portions of Krylov space linear equation solvers. We first present timings from consecutive sweeps over a

Table 1: Three Dimensional Embedding : Mesh Sweeps, Inner Products, SAXPYs : 4K processors

| Grid Size edge | Mesh Sweep MFlops | SAXPY MFlops | Inner Product MFlops |
|---|---|---|---|
| 16 | 23.5 | 131.0 | 14.6 |
| 32 | 46.1 | 226.0 | 81.3 |
| 64 | 64.2 | 233.9 | 185.8 |
| 128 | N.A. | 235.0 | 220.9 |

three dimensional mesh along with timings for the corresponding inner products and SAXPYs. The performance measurements we present here characterize the performance that would arise from the iterative portions of linear solvers employing many simple preconditioners. We then present timing results from a program that performs a sequence of linked matrix mesh sweeps and triangular solves. We argue that the mesh sweeps and triangular solve timings obtained from this benchmark are a fair measure of the timings that would be observed from these procedures were they integrated into an iterative loop of the appropriately preconditioned Krylov solver. As part of this test loop, we also measured the time required to perform inner products in a manner that conformed with data structures and the mapping used for the other two procedures.

The software provided with the CM-2 makes use of the concept of *virtual processors*; one can program the CM-2 so that it appears that there are a larger number of processors than actually exist. The CM-2 software assigns blocks of virtual processors to each real processor. This assignment of multiple virtual processors to each real processor tends to amortize the overhead of transmitting each instruction to the physical processors. In most of the problems we investigate in this paper, increasing the ratio of virtual to real processors also reduces overheads due to communication.

## 2.1 Mesh Sweeps

One can use a very large number of virtual processors in implementing a sweep over a mesh. We examined the performance of a very specialized PARIS program (PARIS is the CM-2 assembly language) which was written for a three dimensional problem on a cube with a seven point operator. The program hence consisted of a sequence of sweeps over a three dimensional mesh with the mesh embedded into a cube of virtual processors with one mesh point assigned to a virtual processor. The cube of virtual processors used by the program had an edge size equal to a power of two. Subject to this constraint, the largest mesh we could embed was one with an edge size of 64. Each iteration of the 1000 carried out took an average of 49 milliseconds. This corresponds to a speed of 64.2 MFlops on the 4K processors. The results obtained from timings for meshes of varying sizes on 4K processors are depicted in Table 1.

In Table 1 we also depict measurements obtained from SAXPYs and inner products over three dimensional domains. All of these results were obtained by timing 100000 consecutive iterations. Because SAXPYs do not require communication, we expect to obtain extremely high performance. For SAXPYs carried out over a cube of virtual processors with edge size 128, we

```
do 100 times

   Mx = y
   Solve Lz = y
   z = inner-product(y,y)

 end do
```

Figure 1: Sweeps over a Mesh

obtained a speed on 4K processors of 235.0 MFlops. The efficiency with which SAXPYs are performed decreases when one uses fewer virtual processors(VP's). The VP ratio specifies the ratio of the total number of virtual processors to the number of physical processors. A cube with edge size 16 has a VP ratio = 1 on a 4K machine. From Table 1 we see that the speed of this computation decreases to 131.0 MFlops, note that this reduction in speed must be unrelated to communication overhead. In Table 1, we also present timings for inner products carried out over a cube of virtual processors with varying edge size. Because the inner products contain a global reduction, we expect to see poorer performance for this operation at relatively low VP ratios. The cost of a global reduction does not increase with problem size, we can thus explain the similarity in the computational rates for the SAXPY and inner product for the highest VP ratios.

## 2.2 Performance from Iterative Loops with Triangular Solves

We next present timing results from a program that performs a sequence of linked matrix mesh sweeps and triangular solves. This set of computations comprises the kernel of several sparse matrix solvers. Let `M` represent a matrix obtained from the uniform discretization of a cube with a seven point template and let `L` represent a lower triangular matrix with the same sparsity structure as `M`. We carried out the test calculation depicted in Figure 1.

This program carried out the matrix vector multiply `Mx = y` by sweeping over a three dimensional mesh. We embedded the three dimensional mesh into a two dimensional gray coded processor lattice. The sparse lower triangular system of equations `Lz = y` was then solved by sweeping over another three dimensional mesh, embedded in a conforming fashion into the same two dimensional processor lattice. The sweep used to solve the sparse triangular system was carried out in a manner that respected the dependencies of the problem. As part of the test loop, we also measured the time required to perform inner products in a manner that conformed with data structures and the mapping used for the other two procedures.

We now describe in more detail how the triangular solve was carried out. In a cube with $n$ points along any edge, $i, j, k$ from 0 to $n-1$ are used to define the position of a point in the cube where $i, j, k$ represent the cartesian coordinates of a point in the 3-D mesh. We can parallelize this three dimensional triangular solve by concurrently solving, for each consecutive $v$, the plane of points satisfying the condition $i + j + k = v$ for $1 \leq v \leq 3n - 2$. Each processor in the lattice

Table 2: Matrix Vector Multiply, Triangular Solve, Inner Product: Optimized: 4K processors

| Grid Edge Size | Inner Product MFlops | Mesh Sweeps MFlops | Triangular Solve MFlops |
|---|---|---|---|
| 16 | 5.5 | 1.7 | 0.5 |
| 32 | 23.4 | 6.9 | 1.8 |
| 48 | 58.2 | 16.9 | 3.9 |
| 64 | 104.9 | 27.1 | 7.0 |
| 128 | 188.9 | 52.6 | 13.1 |

contained, for a given $i$ and $j$, variables corresponding to values of k between 1 and $n$.

For 4K processors Table 2 depicts the timings obtained for various size domains. The timings are averages from 100 iterations. The timings obtained from the cube with edge sizes 128 and 64 were 188.9 and 104.9 MFlops respectively for the inner product, 52.6 and 27.1 MFlops respectively for the sweep over the mesh and 13.1 and 7.0 MFlops respectively for the triangular solve.

We were able to prevent data movement when we followed the mesh sweep by a sparse triangular solve because of the way in which we assigned data to processors. This method of data assignment did have the side effect of requiring us to perform the mesh sweep in $n$ consecutive phases. As one might expect, the use of this embedding does exact a performance penalty when compared to the embedding discussed in Section 2.1. For example we attained a computational speed of 64.2 MFlops for the problem with edge size 64 in Section 2.1, we attain a computational speed of only 27.1 MFlops with the embedding discussed in this section. The conforming inner product also had to be performed in $n$ consecutive phases.

# 3 The Importance of Careful Embedding

## 3.1 Performance Degradation and Irregular Problems

The timings on the CM-2 described above depended heavily on utilizing very specialized programs. The experiments were designed to give the CM-2 virtually every conceivable advantage. The simple, uniform and local nature of the communication in the kernel allowed us to use the optimized NEWS network instead of the more costly general router. In this section of the research note, we explore what can happen to performance when we try to solve problems with non-uniform dependency patterns. We also present some performance measurements obtained when we used the high level language *lisp to code a sweep over a uniform mesh.

We first modified the computational kernel described in section 2.2 so that communication was carried out by data fetch procedure calls through the general router rather than through the NEWS network. Even though the general router was invoked, the mapping of work to processors was actually identical to that described in section 2.2. The timings we obtained with data *fetches* carried out using the general router can be interpreted as best case estimates of what one could expect from a CM-2 executor that did not have access to a-priori information

Table 3: Three Dimensional Mesh Sweep and Solve : News Net v.s. General Router Fetch - Varying Mesh Size: 4K processors

| Edge Size | Mesh Sweep NEWS Mflops | Mesh Sweep Router Mflops | Solve NEWS Mflops | Solve Router Mflops |
|---|---|---|---|---|
| 16 | 1.7 | 0.2 | 0.5 | 0.1 |
| 32 | 6.9 | 0.9 | 1.8 | 0.3 |
| 64 | 27.1 | 3.6 | 7.0 | 1.1 |
| 128 | 52.6 | 5.5 | 13.1 | 1.8 |

Table 4: Performance with 64 by 64 and 128 by 128 Random Mesh

| Grid Size | Type Grid | Mean (ms) | Sqr Root Sample Variance (ms) | Rate Mflops |
|---|---|---|---|---|
| 64 x 64 | NEWS | 1.8 | - | 18.2 |
| 64 x 64 | p = 0.0 | 12.7 | 0.45 | 2.6 |
| 64 x 64 | p = 0.4 | 16.2 | 0.42 | 2.0 |
| 64 x 64 | p = 0.8 | 17.4 | 0.45 | 1.9 |
| 128 x 128 | NEWS | 3.7 | - | 35.0 |
| 128 x 128 | p = 0.0 | 33.5 | 0.77 | 3.9 |
| 128 x 128 | p = 0.4 | 44.5 | 0.50 | 2.9 |
| 128 x 128 | p = 0.8 | 47.9 | 0.57 | 2.7 |

on dependency patterns.

In Table 3 we give comparative computational rates of kernel timings averaged over 100 iterations in which we used either the NEWS network or the the general router. The ratios between the NEWS network and router computational rates range from 6 to 10.

We used a synthetic workload to find out how much CM-2 performance would degrade when the general router was forced to deal with a mesh sweep with irregular communication patterns. A square mesh in which each point was linked to four nearest neighbors was incrementally distorted. Random edges were introduced subject to the constraint that in the new mesh, each point still required information from four other mesh points.

Our workload generator makes the following assumptions: (1) The problem domain consists of a 2-dimensional mesh of points which are numbered using their natural ordering; (2) An edge connects a point to its neighbor with a probability $q \leq 1$. Thus, a random connection is made with probability $p = 1 - q$ (note that when $q = 1$, the model will generate a regular mesh); (3) We use the geometric density function to determine the distance (Manhattan metric) of an edge not connected to its neighboring point. Since we can vary the required communication patterns, these pseudo-random meshes help us evaluate the performance of the machine's network characteristics in a parameterized manner.

Table 4 depicts the performance of sweeps over domains on the CM-2 using different random meshes generated on domains of size 64 by 64 and 128 by 128. We generated meshes for values

of $p$ of 0.0, 0.4 and 0.8. For each value of $p$ we generated 30 different meshes and obtained CM-2 timings for sweeps over each mesh. In this table we present the average CM-2 time for each value of $p$ for both domain sizes examined; we also present the square root of the sample variance. We see a performance degradation of approximately 40 percent between the completely local mesh generated when $p = 0.0$ to the rather irregular mesh generated when $p = 0.8$. For purposes of comparison, we also include the time required using the NEWS network to sweep over the same mesh as is generated using $p = 0.0$. This timing has already been presented in Table 2, in a slightly different context. The three dimensional mesh sweeps in that experiment were carried out by sweeping over consecutive two dimensional planes; we can obtain the desired timing by dividing the mesh sweep times depicted in the table by the grid edge size.

For the 64 and 128 square mesh, when we use the router instead of the NEWS network for the regular ($p = 0.0$) communication pattern, we pay performance penalties of factors of 7 and 9 respectively. As mentioned above, we see an additional 40 percent degradation as $p$ increases from 0.0 to 0.8.

## 3.2 Performance of a Naive Mapping using *lisp

We present some performance measurements obtained when we used the high level language *lisp to code a sweep over a uniform mesh. One of our objectives was to evaluate the ability of the CM-2 to handle even moderately general loops using the CM-2s high level language constructs. The model problem we examine consists of sweeps over square domains of varying sizes with five point templates. The optimized version is explicitly mapped onto the machine in a way that allows us to utilize the CM-2's fast NEWS network for local communication. The other version uses a general router designed to carry out arbitrary patterns of interprocessor communication.

The second version was much more naive. In this version each virtual processor was given the row corresponding to its processor address.The second version corresponds to a naive placement of the matrix, in which no advantage is taken of any geometry inherent in the problem. The programs differed only in the communications calls made. The data structures used were identical.

We see from Table 5 that without very careful mapping and preprocessing, loops that specify very regular computations using general data structures can cause catastrophic degradations in performance. For the 256 by 256 problem on 4K processors the timings for the explicitly mapped NEWS network code corresponded to a speed of 40.0 MFlops, the timings for the router version of the code achieved a speed of 0.5 MFlops. To assure ourselves that the communications overhead was responsible for performance degradation, we performed experiments in which all calls to communications routines were eliminated. From Table 5, we see that without the communications calls, the timings for the two versions of the mesh sweep program were comparable.

Table 5: Mesh Sweep : *lisp: Explicitly mapped News Net v.s. General Router: 4K processors

| Grid Size | News Total (ms) | News Compute Only (ms) | General Router Total (ms) | General Router Compute Only (ms) |
|---|---|---|---|---|
| 64 x 64 | 2.25 | 0.93 | 45.56 | 0.99 |
| 128 x 128 | 4.62 | 1.88 | 189.44 | 1.89 |
| 256 x 256 | 13.11 | 5.46 | 1001.11 | 5.51 |

# 4 Conclusion

In this paper we presented what we might regard as best case timings for the mesh sweeps, sparse triangular solves, and inner products that constitute the iterative portion of certain Krylov space linear solvers. We performed timings on a large three dimensional model problem over a cube shaped domain discretized with a seven point template. As expected we obtained the highest computational rates on SAXPYs and inner products; for a 128 cubed grid the rates were 235 and 221 Mflops respectively. A mesh sweep over a 64 cubed grid was carried out at a rate of 64 Mflops (for this benchmark, the 128 cubed grid did not fit into the available memory). Mapping the mesh sweep in a way that conforms to the sparse triangular solve took a substantial toll on performance; the sweep over the 64 cubed mesh was carried out at a rate of 27 Mflops. We were able to solve a 128 cubed problem using this embedding; the rates for the mesh sweep and the triangular solve were 53 and 13 Mflops respectively.

The performance degraded dramatically when the conforming three dimensional mesh sweep and triangular solve were computed using the general router for communication. For instance, the computational rate of sweeping over the 128 cubed mesh dropped from 53 Mflops to 6 Mflops and the rate of sweeping over a 16 cubed mesh dropped from 1.7 to 0.2 Mflops. We used a synthetic workload to generate meshes with varying degrees of irregularity. As the communication pattern in a mesh became less regular, the computational rate decreased. The most dramatic performance degradation occurred when we compared *lisp mesh sweep codes having optimized versus unoptimized mappings of mesh points to processors. The computational rate of the optimized code on a 256 by 256 mesh sweep was 40.0 Mflops, the computational rate of the unoptimized code was 0.5 Mflops.

The results of our benchmarks clearly demonstrate that the performance obtained on a CM-2 can be exquisitely sensitive to details of mapping and problem structure.

**Acknowedgements:** The authors gratefully acknowledge assistance provided throughout this project by Paul Oppenheimer of Thinking Machines Corporation.

# References

[1] T. F. Chan, C. C. Kuo, and C. Tong. Parallel elliptic preconditioners: Fourier analysis and performance on the connection machine. Report CAM 88-22, UCLA, August 1988.

[2] Paul Concus and Gene H. Golub. Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations. *SIAM Journal on Numerical Analysis*, 10:1103–1120, 1973.

[3] H. Elman. Personal communication. Technical report.

[4] Louis A. Hageman and David M. Young. *Applied Iterative Methods*. Academic Press, New York, 1981.

[5] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix. *Mathematics of Computation*, 31:148–162, 1977.

[6] Richard P. Kendall Todd Dupont and H. H. Rachford Jr. An approximate factorization procedure for solving self-adjoint elliptic difference equations. *SIAM Journal on Numerical Analysis*, 5:559–573, 1968.

# Report Documentation Page

| 1. Report No. | 2. Government Accession No. | 3. Recipient's Catalog No. |
|---|---|---|
| NASA CR-181961<br>ICASE Report No. 89-54 | | |

**4. Title and Subtitle**

KRYLOV METHODS PRECONDITIONED WITH INCOMPLETELY FACTORED MATRICES ON THE CM-2

**5. Report Date**

December 1989

**6. Performing Organization Code**

**7. Author(s)**

Harry Berryman, Joel Saltz, William Gropp, and Ravi Mirchandaney

**8. Performing Organization Report No.**

89-54

**10. Work Unit No.**

505-90-21-01

**9. Performing Organization Name and Address**

Institute for Computer Applications in Science and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23665-5225

**11. Contract or Grant No.**

NAS1-18605

**13. Type of Report and Period Covered**

Contractor Report

**12. Sponsoring Agency Name and Address**

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA23665-5225

**14. Sponsoring Agency Code**

**15. Supplementary Notes**

Langley Technical Monitor:
Richard W. Barnwell

To Appear in J. of Parallel and Distributed Computing

Final Report

**16. Abstract**

In the work presented here, we measured the performance of the components of the key iterative kernel of a preconditioned Krylov space iterative linear system solver. In some sense, these numbers can be regarded as best case timings for these kernels. We timed sweeps over meshes, sparse triangular solves, and inner products on a large three dimensional model problem over a cube shaped domain discretized with a seven point template.

The performance of the CM-2 is highly dependent on the use of very specialized programs. These programs mapped a regular problem domain onto the processor topology in a careful manner and used the optimized local NEWS communications network. We also document rather dramatic deterioration in performance when these ideal conditions no longer apply. A synthetic workload generator was developed to produce and solve a parameterized family of increasingly irregular problems.

| 17. Key Words (Suggested by Author(s)) | 18. Distribution Statement |
|---|---|
| Connection Machine, massively parallel computing, mesh sweep, triangular solve, Krylov | 59 - Mathematical and Computer Sciences (General)<br>64 - Numerical Analysis<br>Unclassified - Unlimited |

| 19. Security Classif. (of this report) | 20. Security Classif. (of this page) | 21. No. of pages | 22. Price |
|---|---|---|---|
| Unclassified | Unclassified | 10 | A02 |